Turk J Elec Engin, VOL.16, NO.2 2008, © TÜBİTAK

# A New Approach Using Temporal Radial Basis Function in Chronological Series

**Mustapha GUEZOURI**
*Signal Image Laboratory, Department of Electronics, Faculty of Electrical Engineering, University of Science and Technology, P.O. Box 1505, El-M'Naouer, Oran, ALGERIA*
*e-mail: mguezouri@yahoo.fr*

**Abstract**

*In this paper, we present an extended form of Radial Basis Function network called Temporal-RBF (T-RBF) network. This extended network can be used in decision rules and classification in Spatio-Temporal domain applications, like speech recognition, economic fluctuations, seismic measurements and robotics applications. We found that such a network complies with relative ease to constraints such as capacity of universal approximation, sensibility of node, local generalisation in receptive field, etc. For an optimal solution based on a probabilistic approach with a minimum of complexity, we propose two TRBF models (1 and 2). Application to the problem of Mackey-Glass time series has revealed that TRBF models are very promising, compared to traditional networks.*

**Key Words:** *Temporal RBF, Classification, Spatio-Temporal, Speech recognition, Robotics applications.*

## 1. Introduction

One limitation of static network is their inability to respond to temporal pattern in the hidden node outputs. To respond to these patterns, the networks must have delay elements within each of its layers. The benefits of internal delays were demonstrated via a dynamic approach in [1], which is especially useful in signal prediction, wherein the input to the network is a time varying signal and the desired output is a prediction of the signal at a fixed lag. Other examples are speech recognition and signal production, when the output autonomously follows a desired trajectory [1].

Day and Davenport [1] introduced back-propagation through time (BPTT) in chaotic signal prediction tasks based on the Mackey-Glass differential delay equation. Lin, Ligomenides & Dayhoff [2] proposed ATDNN [3], which consist of changing the delay time during learning, a method which gave good results in problems of eight and zero form. In problems which require information about the next event, it combines the recurrent neural network (RNN) firstly and adds time delay connections progressively [4]. Another method named Long Short Term Memory (LSTM) applied to time series benchmark problems does not even require RNN at all, because all relevant information about the next event is in fact conveyed by a few recent events contained within a small time window [5].

Our method integrates the time aspect in the RBF neural network. The novelty is that, compared to those methods described above, this approach is not a black box, as we can alter the kernels in sub-

neural networks. In addition, we can apply a Bayesian classifier which introduces the prior-probability, cost of punishment in the case of rejection of misclassification. It also incorporates basic aspects of static RBF in approximation, denseness, uniqueness of interpolation and convergence rate [6]. Other extensions such as moving centres, weighted norm and different types of basis function and multiple scales were also considered. These criteria provide a useful theoretical framework for investigating radial basis function networks and learning algorithms [6]. A variety of approaches for training radial basis function networks have been developed, most of which can be divided into two stages: (i) learning the centres and field receptor in the hidden layer, (ii) learning the connection weights from the hidden layer to the output layer [7]. In this sense, we have proposed our model which integrates the time parameter in the network, in object to resolve some forgotten features in standard model, like memory state, dynamic measures, recalling phases, etc.

We therefore proposed two models: i) first we introduced the delay time only to input neurons, ii) second we inserted delay time in both input and hidden neurones. In sections II and III we described the TRBF neural network and the different models characterizing this approach, mainly the occurrence of time in hidden and input layers disjointedly. In last section, we compared our approach with standard temporal neural networks especially in application to the famous Mackey-Glass chaotic time series.

# 2. T-RBF approach

## 2.1. Definition

The standard RBF can be trained to accomplish pattern recognition tasks with complex non linear boundaries, but are limited to processing static patterns that are fixed rather than time-varying in nature [2]. The Temporal RBF, like ATDNN, LSTM, etc., have been proposed to overcome this limitation. Networks with this capability can play an important role in applications that are dynamic and naturally time-varying.

Also, like classical RBF, their goal is to approximate a desired approximation by a collection of functions, named kernels [8, 9]. A kernel is characterized by a centre $Ci$ and receptive field $r$, and can be chosen by $k$-means clustering or vector quantification.

All these parameters can be taken in account to introduce the Bayesian probabilistic classifier and prior knowledge about the problem. Moreover, we can combine this approach with other techniques like Hidden Markov Models (HMM) by using the generating probabilities. In general, the temporal discrimination function of class $K$ is written in the following form [2]:

$$y_k(t_n) = \sum_{j=l}^{ml} w_{jk}\varphi(\sum_{l=0}^{pl} w_j(l)x(t_n - l) + b_j) + b_0. \quad (1)$$

Next, we describe the network architecture and learning algorithm.

## 2.2. Network architecture

We propose the following architecture: an input layer with $n$ measures, including a delay block; a hidden layer with $h$ nodes, each including a delay block; and an output layer with only one node (see Figure 1 and Figure 2).

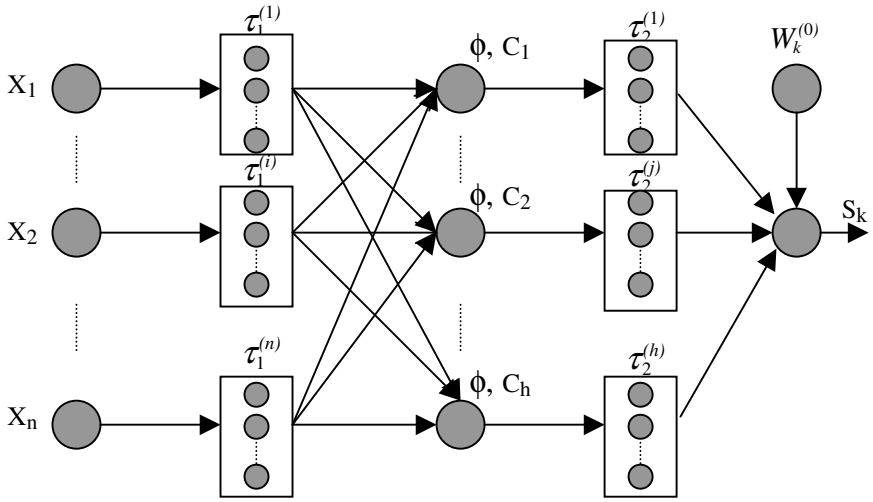


**Figure 1.** General model for T-RBF.

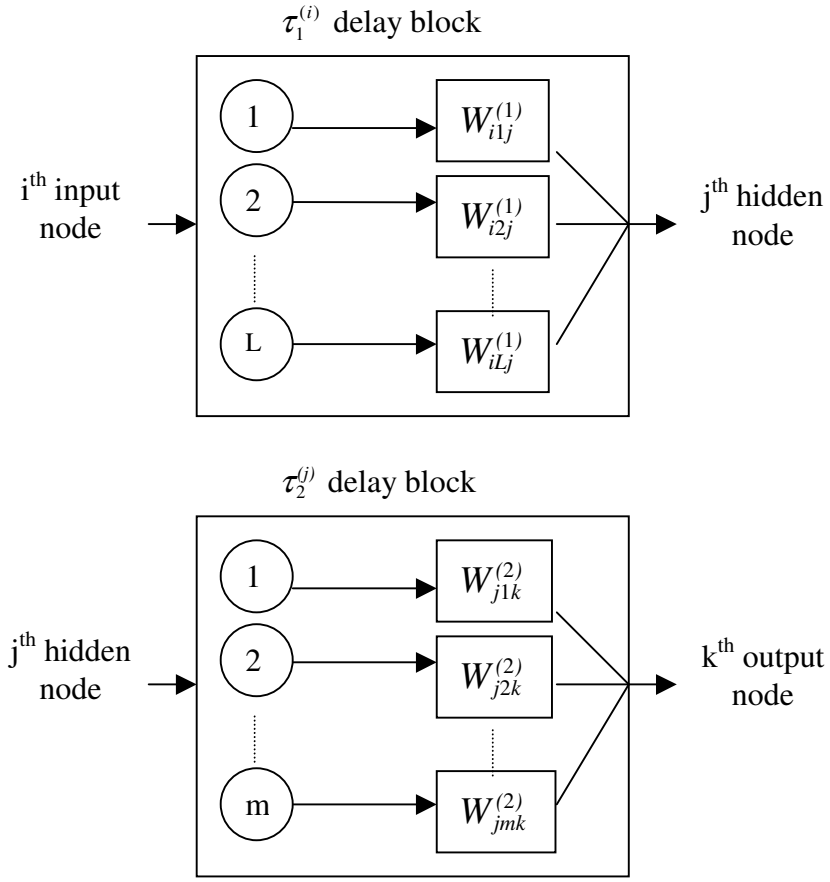


**Figure 2.** Representation of $\tau_1^{(i)}$ and $\tau_2^{(j)}$ delay blocks. $\tau_1^{(i)}$ corresponds to the delay block of the input measure for the $i^{th}$ node, and $\tau_2^{(j)}$ corresponds to the delay block of the hidden activation function for the $j^{th}$ node.

Now we define the output function as:

$$S_k = \varphi\left(W_k^{(0)} + \sum_{j=1}^{h} \sum_{p=1}^{\dim(\tau_2^j)} Y_{jp} W_{jpk}^{(12)}\right), \tag{2}$$

where $\varphi$ is the output activation function, which takes a linear or sigmoid form, $W_k^{(0)}$ is the bias and $W_{jpk}^{(12)}$ represent the weights between the hidden and output layers.

The hidden activation function is

$$Y_{jp} = \sum_{i=1}^{n} \sum_{q=1}^{\dim(\tau_1^i)} W_{iqj}^{(1)} \phi\left(\|X_{iq} - C_{jp}\|\right), \tag{3}$$

where $\phi$ is a radial basis function, $W_{iqj}^{(1)}$ represent the weights between the input and hidden layers, $C_{jp}$ represent the centroid value for the $p^{th}$ order of the $j^{th}$ hidden node and $X_{iq}$ represent the input measure value for the $q^{th}$ order of the $i^{th}$ input node.

## 2.3. Network Unfolding Algorithm network

The difficulty in the problem is how to estimate the activation functions for each delay block elements in the hidden layer. One solution is reducing this delay block to only one delay. For this object we apply the Network Unfolding Algorithm (NUA) developed by Lin [10]. This algorithm allows us to simplify the architecture of Figure 1 and reduces the complexity of computation; hence we can apply the learning algorithm. For this purpose, we must apply two phases.

**Phase 1**

This phase involves the following four steps.

- **Step 1:** unfold inputs.

  For each hidden node $j$, do in parallel:

  For each input node $i$, do in parallel:

  - duplicate the new input nodes and spread these nodes horizontally next to each original input node.
  - move the original connections to next input nodes, respectively, and retain the weight and time-delay on each connection.

- **Step 2:** re-adjust input time lag:

  For each hidden node $j$ do in parallel:

  - remove the time-delays between input and hidden units.
  - set the input values as the signal value vector Transmitted from $i^{th}$ input node to $j$.

- **Step 3**: Unfold hidden nodes.

  For each output node $k$ do in parallel:

  For each hidden node $j$ do in parallel:

– duplicate the new input nodes and spread these nodes horizontally next to each original hidden node.
– move the original connections to new hidden nodes correspondingly and retain the weight and time-delay on each connection.
– For each newly created node do in parallel copy the whole branch which associates with the original hidden node in step2, and then connect to that new node as its branch and retain the weights.

- **Step 4:** Re-adjust input time lag as follows.

  For each hidden node $j$ and it's newly created node from step 3, do in parallel:

  – remove the associated time delays between hidden and outputs units.
  – re-adjust the input node time lag such that each the input node of each branch takes the signal value vector delayed by $\tau_2$ and transmitted from node i to j.

To explain more and ease the understanding of the algorithm, we apply the network unfolding algorithm (NUA) on the scheme represented in Figure 3, and we obtained a new unfolded scheme, shown on Figure 4.

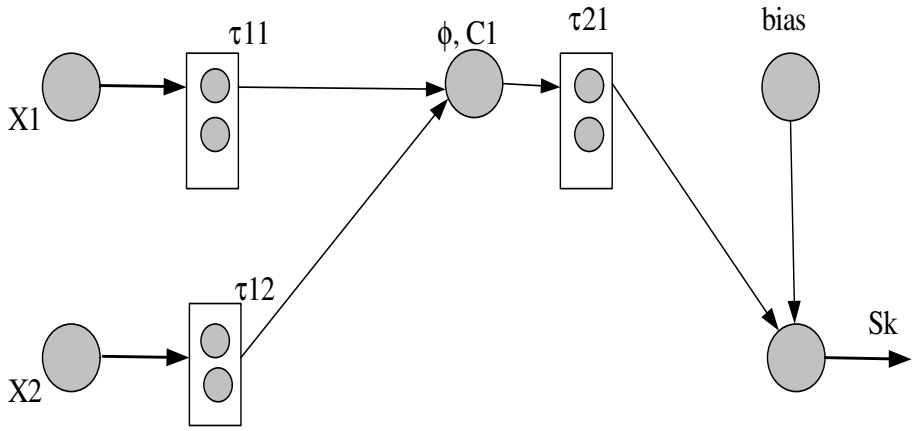


**Figure 3.** Initial network with two blocks ($\tau_1$, $\tau_2$).

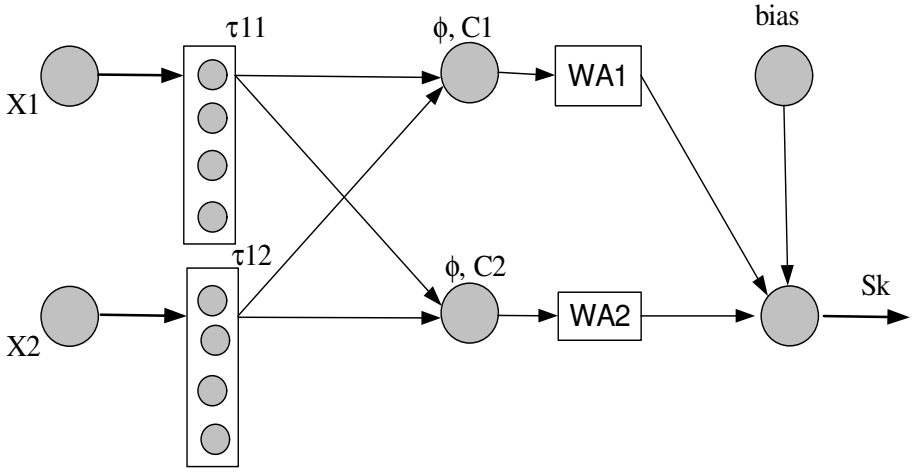


**Figure 4.** Deletion of delay block 2 by unfolding.

**Phase 2**

This phase is applied to fuse two hidden neurones in one hidden neurone, in order to represent only one centre at each iteration (see the transition between Figure 4 and Figure 5). This phase allows us to apply the Orthogonal Least Square (OLS) learning algorithm [6] described in next section. The OLS method is an incremental learning technique that iteratively creates a new node representing a new cluster until obtaining a sufficient number of kernels.

In summary, phases 1 and 2 yield the following tasks:

- The connection between the hidden layer and the output layer is based only on the weights, without time delay.
- The connection between the input layer and the hidden layer is based only on time delay, without weights.

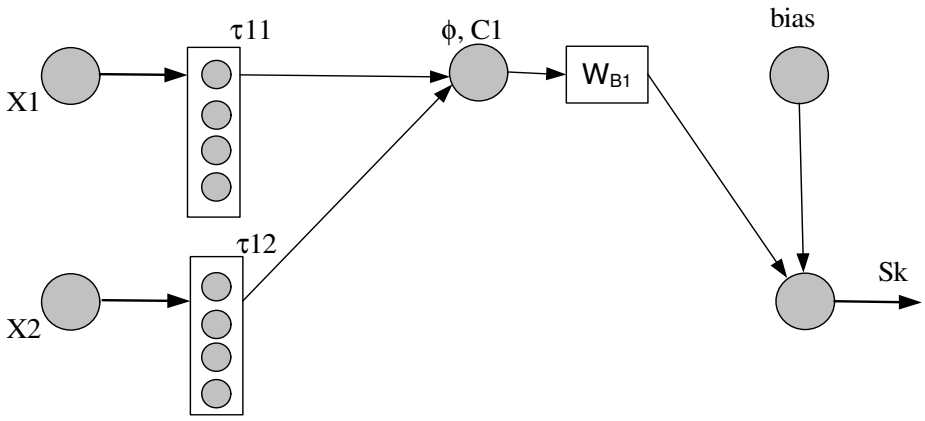


**Figure 5.** Resulting network after fusion.

## 2.4. Learning algorithm

The Orthogonal Least Square (OLS) algorithm has been applied as the learning step. We suppose that the kernel function $\phi$ is fixed and have the same form in all hidden neurones. Therefore this algorithm allows us to perform incremental learning [8]. This algorithm is preferred over others since it respects the following points:

- OLS provides a linear separation between the input layer and the hidden layer and creates the hidden neurones automatically [11], with application of Gram-Schmidt orthogonalisation.
- Second, the synaptic weights between hidden and output layers are calculated via the least squares method.

We consider the RBF network as a particular case of a linear regression model [6], defined as

$$d(t) = \sum_{i=l}^{P_i\phi_i+\varepsilon(t)} . \tag{4}$$

where $d(t)$ is the desired output at time $t$; $\phi_i$ are the search parameters; $\varepsilon(t)$ is the approximation error of $d(t)$; and $P_i(t) = P_i(x(t))$ the fixed functions of $x(t)$. The calculated function by the RBF network is the same described in Equation (4), the analogy is:

$$d = P \cdot \phi + E. \tag{5}$$

We employ the following notation: $d$ is the desired output vector; $d = [d(1),\ldots,d(N)]^T$; $N$ is the number of examples of learning basis; $M$ corresponds to the initial number of centres; $P$ is the matrix of the hidden layer outputs; $P = [P1,\ldots,P_M]$; $P_i$ is the vector of the $i^{th}$ hidden cell; $P_i = [P_i(1)\ ,\ldots,\ P_i(N)]^T$; $\theta$ is the weights of output layer; $\theta = [\theta_1,\ldots,\theta_M]$. $E$ corresponds to the errors between the estimated and desired outputs: $E = [\varepsilon(1),\ldots,\varepsilon(N)]^T$. The resolution of systems equation (5) is a trivial problem. The solution vector $\theta$ can be defined by the mean square method.

The origin of the OLS method resides in transformation of the P matrix to a matrix with orthogonal columns, one by one. The orthogonalisation of columns $P_i$ can be obtained by the decomposition of $P$ matrix in two matrices $W$ and $A$:

$$P = W \cdot A, \tag{6}$$

where $W$ is of size $N \times M$, is the orthogonal image of matrix $P$; $A$ is of size $M \times M$, is a superior triangular matrix, contains the orthogonalisation coefficients.

The $A$ matrix is defined as follows:

$$A = \begin{matrix} 1 & \alpha_{1,2} & & & & \alpha_{1,M} \\ 0 & 1 & & & & \alpha_{2,M} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & 1 & \alpha_{M-2,M-1} & \cdots \\ 0 & 0 & & 0 & 1 & \alpha_{M-1,M} \\ 0 & 0 & & 0 & 0 & 1 \end{matrix}$$

The space generated by the $P_i$ vectors is the same space obtained by the $W_i$ vectors, and the systems equation (5) can be rewritten in the new form

$$d = W * G + E. \tag{7}$$

Here $G = A \cdot \theta$ is the searched solution. Note that

$$H = W^T \cdot WE. \tag{8}$$

Since the columns of $W$ matrix are orthogonal one by one, $H$ is a diagonal matrix with $h_i$ elements and we have

$$h_i = w_i^t w_i = \sum_{j=l}^{w_{ij}w_{ji}}, l \le i \le M. \tag{9}$$

This propriety makes useful the OLS method, for the reason: the orthogonal solution G is calculated by:

$$G = H^{-1} \cdot W^t d. \tag{10}$$

We can calculate $G_i$ by

$$g_i = \frac{w_i^t d}{w_i^t w_i}, l \le i \le M. \tag{11}$$

This signifies that elements $g_i$ of orthogonal solution $G$ are dependant only on column $w_i$; in other words they depend on orthogonal image of calculated output for each centre. This part defines the quotient of approximation error reduction, introduced by each $w_i$ vector and can be expressed by:

$$[err]_i = \frac{G_i^2 w_i^t w_i}{d^t d}, l \leq i \leq M \tag{12}$$

This equation is used to construct the RBF network in iterative manner. Beginning with initial set of $M$ centres, the network is constructed to each iteration, by adding a centre which have the $[err]_i$ maximal, and we take the correspondent $G_i$. For each iteration, we calculate the $A$ and $W$ elements by:

$$\alpha_{j,k}^i = \frac{w_j^t * pi}{w_j^t * wj} \tag{13}$$

$$w_k^i = Pt - \sum_{j=l}^{j=k-l} \alpha_{j,k}^i * wj \tag{14}$$

We use the Akaike criterion to limit the iterations:

$$l - \sum_{i=l}^{M} err_i > \varepsilon. \tag{15}$$

In the end of iterations, we calculate $\theta_i$ (the synaptic weights) by the following system:

$$G = A \cdot \theta. \tag{16}$$

# 3. T-RBF models

## 3.1. Model 1

In this model, we consider the network only with delay $\tau_1$, meaning we take into account 1 measure of delay from the input layer with simply one connection between hidden and output layers, for each hidden neuron.

## 3.2. Model 2

In this case, we considered the neural network with two delay blocks $\tau 1$ and $\tau 2$. In the first block $\tau 1$, we take into account "$L$" measures delayed entry for each entry node; and for the second block $\tau 2$, we take "$m$" delayed activations for each hidden node.

# 4. Experimentation

## 4.1. Application of Mackey-Glass series

The Mackey-Glass series (1977) is a case of typical dynamic system [4], which describes the production of white blood cells. In this system, we can apply our approach to estimate the actual states based on previous measures. The system of Mackey-Glass can be generated from the delay differential equation

$$\frac{dx(t)}{dt} = \frac{ax(t-\tau)}{l + x^c(t-\tau)} - bx(t). \tag{17}$$

With $a = 0.2$, $b = 0.1$, $c = 10$, $\tau = 17$ and the initialisation is $x(t)$=0.8 for $t <$=17 in our application, for $\tau >$17 the series become chaotic; and for $\tau = 17$ is a quasi periodic series [4][5][2].

## 4.2. Data basis

In object to simulate the process, we have constructed a basis of 300 (examples and their corresponding targets), each example being represented by $< x(t_i),\ x(t_i - \tau) >$ and its corresponding target is $< x(t_i + 1) >$, where

$$x(t_{i+1}) = x(t_i) + \frac{dx(t)}{dt} \mid t = t_i. \tag{18}$$

## 4.3. Configuration of parameters

We describe now three types of configuration parameters used in this investigation.

- **Size of input and hidden delays block**

We have used model 2 of TRBF, the length of delay block $\tau_1$ is 1 and the length of delay block $\tau_2$ is 2. We need one input layer node, one output layer node and sufficiently some hidden layer nodes.

- **Akaike criterion**

When using the OLS algorithm, the one must use the criterion where *err* is fixed to a value $err \leq 0.001$ in the process. The number of iterations is limited based on quadratic error. In turn this determines the complexity of the hidden layer (viz. number of kernel functions characterized by theirs centres and receptor field). We have obtained three hidden nodes in the TRBF case.

- **Gaussian kernel**

This kernel function is characterised its asymptotic properties, and for its accuracy in learning and generalisation phases. It is presented by the cluster (meaning centre) and the spreading deviation.

## 4.4. Obtained results

- **Mean Square Error**

To show the veracity and the good performance of the results, we have computed the Mean Square Error (MSE) between calculated outputs from TRBF and targets from the learning base. We have obtained the MSE equal to 0.02.

- **Comparison**

We show the results obtained by the application of analytic differential equation towards the results obtained by TRBF approach, we see the different cases in the following figures 6 and 7. For example in figure 6 the curve of $x(t)$ towards $x(t\text{-}1)$ in our approach falls approximately on the same position of curve obtained by analytical differential equations.

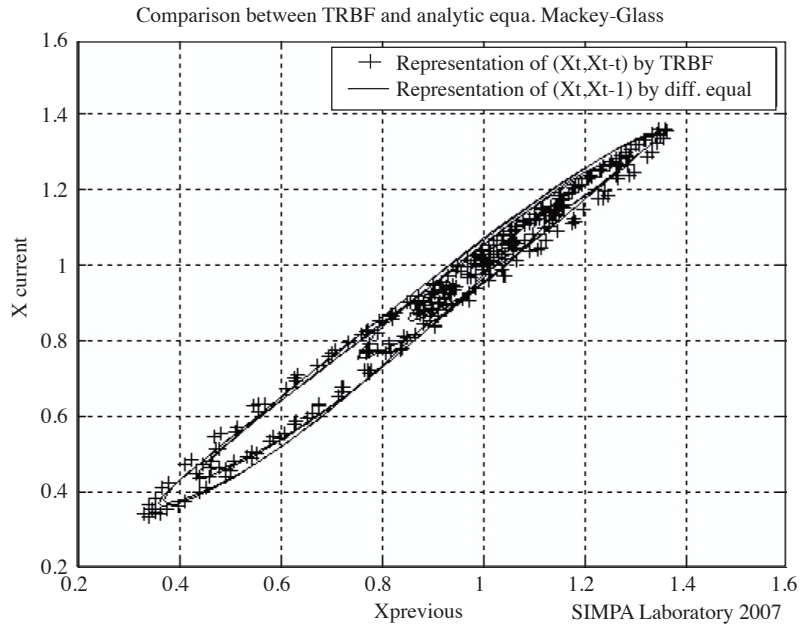


**Figure 6.** State space (current and previous states).

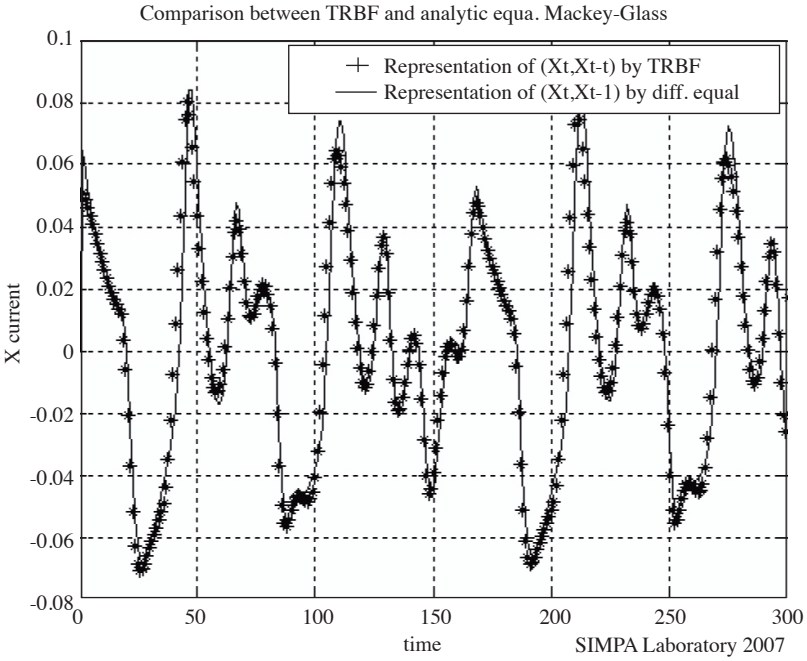


**Figure 7.** Strange attractors on state space.

We conclude this work with a comparative table which summarise the results obtained by different temporal neural network approaches [5] against our model which have given a good results with a minimum of parameters.

We define the abstract terms given in Table 1:

- ATNN: Adaptive temporal neural network;
- SOM: Self Organising Map;

- MLP: Multi Layers Perceptron;
- AMB: An improved Memory Based regression;
- LSTM: Local Short Term Memory;
- TRBF: Temporal Radial Basis Function.

We haven't develop these approaches except the TRBF. The different results are reported from different papers referenced in bibliography. Our object is to compare our results to others mainly on complexity and normalised means square error NMSE.

**Table 1.** Position of TRBF among different temporal methods. T represents the time delay; units: the number of nodes.

| Temporal Methods | Units | Parameters | NMSE | | |
|---|---|---|---|---|---|
| | | | $T = 2$ | $T = 6$ | $T = 84$ |
| ATNN | 20 | 120 | - | 0.005 | - |
| SOM | - | 10×10 | - | 0.013 | 0.06 |
| MLP | 4 | 25150 | - | 0.0511 | 0.46 |
| AMB | - | - | - | - | 0.054 |
| LSTM | 4 | 113 | 0.0214 | 0.1184 | 0.47 |
| TRBF | 5 | 9 | 0.0198 | 0.005 | - |

## 5. Conclusion

Inspecting the comparative table and all figures we can say that the spatio-temporal estimator based on temporal radial basis function is effective and accurate in time modelisation, with minumum complexity. The advantage of this work is to show that a complicated problem like the Mackey-Glass time series prediction can be recognised by such a simple TRBF architecture. Thus we conclude from this study that the TRBF is a best tool to tackle temporal signal processing and estimation problems.

The network accomplishes the identification of background model of Mackey-Glass delay differential equation again with high accuracy (NMSE =0.0198). This proposed neural architecture can be useful and promising in biomedical prediction tasks, recognition of trajectories from moving targets, motion of visual images, robotic application and speech recognition.

## References

[1] S.P. Day, M.R. Davenport, Continuous-Time temporal Back-Propagation with adaptable Time Delays. IEEE Transaction on Neural Networks 1993; 4(12):348–354.

[2] D.T. Lin, The adaptive Time delay Neural Network Characterization and Application to Pattern Recognition, Prediction and signal processing. Thesis Report PHD. University of Maryland, 1994.

[3] C. Wohler, J.K. Anlauf, Real time object recognition on image sequences with adaptable time delay Neural Network Algorithm -application to autonomous vehicles. Image and Vision, Computing Journal 2001; 19(9–10): 593–618.

[4] R. Bonne, M. Cruciano, J.P.A. De Beauville, An algorithm for the addition of time-delayed connections to recurrent neural network. ESANN'2000 proceeding, Bruge (Belgium); 293–298.

[5] F. Gers, D. Eck, J. Schmidhuber, Applying LSTM to time series predictable through time window approaches. Technical Report; IDSIA-IDSIA-22-00. Institute Dalle Molle di studi Sull intelligenza artificiale, Galleria2, Switzerland, 1999.

[6] N.B. Karyianis, Reformulated radial basis function neural network trained by gradient descent. IEEE transaction on Neural Network may 1999; 10(3): 657–669.

[7] S. Haykin, Neural Networks a comprehensive foundation. Prentice Hall Upper Saddle River, New Jersey, 1999.

[8] G. Zheng, S. Billings, Radial basis function network configuration using mutual information and the Orthogonal Least Squares algorithms. Neural Network 1996; 9(6): 1619–1637.

[9] Z. Mekkakia, L. Mesbahi, M. Lakehal, Initialising of RBF centers by SOM Algorithm. MS'2002, International Conference on modeling and simulation in technical and social sciences, Girona, Catalonia, Spain 25–27 june 2002; 717–723.

[10] D.T. Lin, J.E. Dayhoff, Network Unfolding Algorithm and universal spatio-temporal function approximation. Technical Research Report TR95-6. Institute for system research ISR; University of Maryland; College Park MD.

[11] M.K. Titsias, A.C. Likas, Shared kernel model for class Conditional density Estimation. IEEE transaction on Neural Network September 2001; 12(5): 987–996.